MULTI-PERSON 2D HUMAN POSE ESTIMATION A BENCHMARK FOR REAL-TIME APPLICATIONS

Tomislav Prusina[1*] – Juraj Benić[1] – Domagoj Ševerdija – Domagoj Matijević[1]

[1]School of Applied Mathematics and Computer Science, University of Osijek, Trg Lj. Gaja 6, 31000 Osijek, Croatia

ARTICLE INFO

Article history:
Received: 4. 3. 2026.
Received in revised form: 25. 3. 2026.
Accepted: 25. 7. 2026.

Keywords:
Benchmark
Human pose estimation
MoveNet
OpenPose
Sapiens
RTMO
RTMPose
YOLO

DOI: 10.30765/er.3423

Abstract:

Human pose estimation (HPE) is a critical component of computer vision, enabling real-time applications across various fields. This study benchmarks six state-of-the-art frameworks: OpenPose, YOLO, RTMO, RTMPose, Sapiens, and MoveNet in the context of 2D multi-person pose estimation in videos for real-time applications. Using the Panoptic dataset and standardized metrics, including Object Keypoint Similarity (OKS), Average Precision (AP), and Average Recall (AR), we evaluate their accuracy, speed, and hardware efficiency under realistic conditions such as occlusion and crowded scenes. Our analysis highlights the strengths and limitations of each framework, providing valuable insights to help practitioners select and deploy reliable HPE solutions in real-world applications.

ACCEPTED

1. Introduction

Human pose estimation (HPE) is a cornerstone of computer vision, enabling machines to interpret human actions and interactions in various domains such as sports analytics, augmented reality, industrial automation, and others. In sports analytics, for instance, HPE plays a pivotal role in systems like the semi-automated offside technology (SAOT) [1] recently introduced in professional soccer. By tracking the precise positions of players and the ball in real time, these systems utilize pose estimation to accurately determine the relative positioning of body parts, such as feet and shoulders, when assessing offside situations. Such advancements underscore the growing reliance on HPE for real-time decision-making in high-stakes environments. Beyond sports, the demand for multi-person, real-time pose estimation has surged across various domains, driven by applications in environments where quick and accurate decisions are critical. In addition, real-world settings present unique challenges, including occlusion, motion blur, diverse body orientations, and varying environmental conditions, making HPE a complex and continuously evolving research area.

While advancements in deep learning have significantly improved the accuracy and efficiency of HPE methods, evaluating their real-world performance remains a persistent challenge. Most benchmarks are conducted by the developers of these solutions, often under idealized conditions or using constrained datasets, leaving practitioners with limited insight into their applicability in specific use cases. Furthermore, discrepancies in metrics, datasets, and implementation practices across studies create additional barriers to comparing and adopting these methods in realistic scenarios. Commonly used resources span both image and video-based settings: LSP, MPII, Panoptic, COCO, CrowdPose, and the PoseTrack series (2017, 2018, 2021). The PoseTrack project introduced a large-scale benchmark for video-based multi-person pose estimation and articulated tracking, addressing inconsistencies in body-pose trajectories by providing a new annotated dataset, evaluation server, and performance analysis, though the link to the evaluation server https://posetrack.net/ at the time of the writing of this text was not available.

Several surveys have comprehensively reviewed the development of 2D-HPE methods. Munea et al. [2] classify HPE approaches into single-person and multi-person paradigms, emphasizing top-down and bottom-

* Corresponding author
E-mail address: tprusina@mathos.hr

up strategies. Their review highlights the transformative role of deep learning in achieving state-of-the-art accuracy and introduces a taxonomy for evaluating model architectures and metrics. Similarly, Chen et al. [3] provide a detailed analysis of network design, training refinement, and post-processing strategies, offering a broad perspective on methodologies and datasets. Other noteworthy contributions to the field include Dang et al. [4], Kulkarni et al. [5] and Lan et al. [6], which delve into specific academic challenges and advancements in pose estimation.

Despite these valuable contributions, a gap remains in evaluating HPE methods under real-time constraints. Existing reviews and benchmarks rarely account for the unique demands of real-world applications, such as handling multi-person scenarios or occlusions while operating within stringent hardware and timing limitations. Our analysis focuses on solutions capable of processing video data, a critical requirement in many realistic scenarios.

Rather than presenting another taxonomy of algorithms, we evaluate several prominent methods identified as strong candidates for real-world applications: OpenPose [7], YOLO [8], RTMO [9], RTMPose [10], Sapiens [11], and MoveNet [12]. We also evaluated popular Google's MediaPipe for its multi-person option, but found its single-person-centric architecture and grouping heuristic struggled badly in multi-person scenes. Consequently, MediaPipe was not included in our benchmarking suite.

As an independent and objective third-party team with no prior involvement in the development or implementation of these frameworks, we benchmark these state-of-the-art solutions within the context of 2D multi-person pose estimation, specifically targeting real-time applications. To ensure fairness and real-world relevance, we use standardized accuracy metrics such as Object Keypoint Similarity (OKS), Average Precision (AP), and Average Recall (AR), and measure inference speed in frames-per-second (FPS) on GPU hardware. Evaluations are conducted under realistic conditions, including challenging scenarios such as occlusion and high-density scenes, to reveal the practical latency-accuracy trade-offs critical for deployment.

We evaluate each method on a completely unseen dataset, excluded from both training and prior benchmarks, to measure true generalisation under domain shift (e.g., new camera setups and motion patterns) that random splits often miss. Using the challenging Panoptic [13] video dataset, we reveal performance drops that stay hidden when models are tested only on held-out training data. Our results are context-specific: rather than declaring any method universally superior, we analyse each approach's strengths and limitations for multi-person 2D pose estimation in real-world videos. For more details and to access the code, please visit the project's GitHub repository https://github.com/JurajBenic/HPEBench2D.

REJECTED

2. Preliminaries

2D human pose estimation (HPE) is a computer vision problem that involves detecting and localizing key anatomical landmarks (keypoints or joints) of one or more human bodies in an image or video (we consider a video to be a sequence of images). The objective is to predict the spatial positions of these predefined keypoints in the image's given 2D-pixel coordinate system.

Formally, let $I \in \mathbb{R}^{H\times W\times C}$ denote an image, where H, W, and C are the height, width, and number of channels (e.g., RGB). Each image may contain multiple persons (poses). Specifically, let $P^{(j)}$, $j = 1, \ldots, N$ represent the N human poses in the image. Each pose $P^{(j)}$ is defined as a sequence of K keypoints (e.g., shoulders, elbows, wrists, hips, etc.)

$$P^{(j)} = \{(x_i, y_i), i = 1, \ldots, K\}, \quad (1)$$

and the output $\hat{P}^{(j)}$ of a typical 2D pose estimation algorithm

$$\hat{P}^{(j)} = \{(x_i, y_i, p_i), i = 1, \ldots, K\}, \quad (2)$$

where x_i and y_i are the 2D coordinates and p_i is the confidence score for the i-th keypoint of person j.

3. Methods

2D-HPE results vary across many public benchmarks, which use different datasets and metrics. To reduce this inconsistency, we benchmark RTMO, RTMPose, and YOLO11x as representative real-time methods relevant to industrial settings. We also include OpenPose as a widely used early baseline, Sapiens (Meta Reality Labs) as a recent method reporting substantial gains over prior work, and MoveNet as Google's ultra-fast model designed for mobile and edge devices. We provide a high-level description of each method and a summary in Table 1. Additionally, in Table 2 we provide an overview of the datasets used to train these

methods. As this work focuses on benchmarking rather than surveying, readers seeking a more detailed understanding are encouraged to refer to the original papers.

Table 1. – Methods evaluated in this work and their possibilities

Method	Pose	Hand	Face	Bounding-box	Easy to install
OpenPose	main model	separate model	separate model	no	no
RTMO	main model	not supported	not supported	yes	no
RTMPose	main model	separate model	separate model	yes	no
Sapiens	main model	main model	main model	no	no
YOLO11x	main model	separate model	not supported	yes	yes
MoveNet	main model	not supported	not supported	yes	yes

ACCEPTED

3.1. OpenPose

OpenPose is a widely used open-source library for real-time, multi-person 2D pose estimation. It uses a bottom-up, multi-stage CNN (i.e., it first detects body keypoints across the whole image and then groups them into individual people) with two outputs: Part Confidence Maps (PCM) for keypoint locations and Part Affinity Fields (PAF) for linking keypoints into limbs. Final poses are formed by matching keypoints using a greedy bipartite procedure. The COCO pose model used here was downloaded from Kaggle and has ~52.3M parameters. OpenPose is no longer maintained, so installation is often difficult and typically requires building from source with CMake and multiple dependencies, which can lead to compatibility issues. OpenPose can also be used via OpenCV, but this often defaults to CPU execution unless OpenCV is rebuilt with GPU support. In addition, when using OpenCV's DNN module, keypoint grouping into individual poses must be implemented manually or handled with third-party code.

3.2. RTMO

RTMO (Real-Time Multi-Person Object) is a fast model for real-time multi-person pose estimation that extends the YOLO architecture with coordinate classification. Instead of running pose estimation separately for each detected person, it performs a single-pass prediction over the full image. Using dual convolution blocks at each spatial level, RTMO produces per-grid confidence scores and pose features, which are then used to infer bounding boxes, keypoint coordinates, and visibility. This bottom-up design enables high speed, especially in crowded scenes. In this work, we use RTMO-L (rtmo-l_16xb16-600e_body7-640x640) with ~44.8M parameters, trained on COCO, CrowdPose, AIC, MPII, JHMDB, Halpe, and PoseTrack18. The model provides ready-to-use inference scripts, but installation is moderately complex because it relies on OpenMMLab's MMPose and requires specific, compatible versions of mmengine, mmcv, and mmdet. In practice, this often means installing older mmcv/mmdet releases and using Python 3.8.

3.3. RTMPose

RTMPose is a series of real-time models for 2D multi-person pose estimation designed to balance speed, accuracy, and deployment efficiency, addressing the challenges of real-time performance on devices with limited computing power. It adopts a top-down approach with real-time detectors, leveraging CSPNeXt as the backbone for efficient feature extraction and a SimCC-based keypoint localization algorithm, which reduces computational costs compared to heatmap-based methods. The models are available in multiple sizes (t/s/m/l/x) to meet diverse performance and speed requirements across various hardware platforms. We report results using RTMPose-L (rtmpose-l_8xb256-420e_body8-384x288), which has ~27.79M parameters and provides high-precision pose estimation while maintaining real-time throughput on our test hardware, making it suitable for benchmarking where both accuracy and efficiency are required. RTMPose-L is trained on AIC, MS COCO, CrowdPose, MPII, sub-JHMDB, Halpe, PoseTrack18, and OCHuman, supporting robust performance across diverse settings. It uses the CSPNeXt-l* backbone (cspnext-l_udp-body7_210e-384x288-b15bc30d_20230504; ~32.44M parameters), bringing the total for our method to ~60.23M parameters. Our

implementation matches RTMO, differing only in the inference-script alias, since both are provided within the MMPose toolbox.

3.4. Sapiens

Sapiens is a family of vision transformer models for four human-centric tasks: 2D pose estimation, body-part segmentation, depth estimation, and surface normal prediction. It comes in multiple sizes (0.3B/0.6B/1B/2B) and is pretrained with Masked Autoencoders on Humans-300M, a curated set of 300M in-the-wild human images. For pose estimation, it uses a dense 308-keypoint skeleton and supports 28 segmentation classes, with high-resolution synthetic data for depth and surface normals. We report results for Sapiens-0.6B, the largest variant that ran on our hardware. Compared to lightweight pose-only models (e.g., RTMPose/RTMO/OpenPose), Sapiens is substantially larger and less efficient, but offers broader capabilities, including detailed pose plus hands, feet, and facial keypoints (308 total). It follows a top-down pipeline using RTMDet-m (rtmdet_m_640-8xb32_coco-person_no_nms.py) as the person detector; we include it to assess whether this added task diversity justifies the higher compute cost for 2D pose estimation benchmarks. Installation is comparatively complex because it depends on OpenMMLab components (e.g., mmengine, mmcv, mmdet) and a compatible off-the-shelf detector; at the time of writing, an open GitHub issue reported installation failures due to module incompatibilities.

3.5. YOLO11x

YOLO11x is the largest model of the popular YOLO (You Only Look Once) family, adapted for pose estimation tasks, with approximately 56.9 million parameters. Leveraging the efficient object detection capabilities of YOLO, it employs a top-down approach to detect human poses, aiming to balance high accuracy and real-time performance across diverse environments. YOLO11x is trained on the COCO dataset and stands out as the simplest method among those used in this work. Unlike other methods, YOLO11x does not require extensive tweaking or the installation of additional modules. It can be easily installed using pip and is ready for use out-of-the-box.

3.6. MoveNet

MoveNet is Google's pose detection model designed for real-time applications on mobile and edge devices. MoveNet is a bottom-up pose estimation model that directly predicts the positions of 17 human keypoints for 6 people without requiring an explicit person detection step. For this work, we use MoveNet MultiPose Lightning, the only MoveNet variant that natively supports multi-person pose estimation (up to six individuals per frame) while maintaining real-time performance. With ~4.8M parameters, it is substantially lighter than the other methods evaluated. It is pre-trained on COCO and distributed via TensorFlow Hub, enabling straightforward installation and deployment with minimal setup and out-of-the-box pose estimation functionality. The selected benchmark did not contain frames with more than six annotated persons, so MoveNet's architectural limit did not require any special post-processing in our evaluation.

Table 2. – Datasets used to train evaluated methods

Dataset	Year	Images/Videos	Multi Pose
LSP	2010	yes/no	no
MPII	2014	yes/yes	yes
Panoptic	2015	yes/yes	yes
COCO	2017	yes/no	yes
PoseTrack 17	2017	no/yes	yes
PoseTrack 18	2018	no/yes	yes
CrowdPose	2019	yes/no	yes
PoseTrack 21	2021	no/yes	yes

4. Panoptic dataset

The Panoptic dataset is uniquely suited for benchmarking the proposed models, as it is the only dataset on which none of the evaluated methods have been trained or tested. Shown in Table 3, the dataset comprises 84 sequences, amounting to a total duration of 11 hours, 29 minutes, and 15 seconds. Each sequence is recorded using an extensive camera setup, including 480 VGA cameras, 31 HD cameras (1920x1080@30fps), and 10

ACCEPTED

Kinects [13]. The sequences are captured within a dome, ensuring that all keypoints remain visible at all times in the 3D space. This setup allows for the generation of a 3D human pose representation. This makes the dataset particularly valuable for assessing the accuracy and robustness of the proposed methods. We restrict our analysis to the 28 representative sequences listed in Table 2, comprising 7 scenarios with 4 sequences each, totaling 121.33 minutes at 30 FPS in HD resolution. These carefully selected sequences provide a suitable benchmark for evaluating the proposed methods.

Table 3. - Used videos from Panoptic dataset for benchmarking

Scenario Name	Alias	Category	Duration	Number of People	Camera ID's
160906_band2	band	Musical instruments	300 s	3 to 4	0, 6, 13, 22
160906_ian1	ian	Toddler	120 s	1 to 2	0, 2, 12, 22
160906_pizza1	pizza	Others	300 s	4 to 6	0, 5, 15, 25
170221_haggling_m3	haggling	Social games (haggling)	260 s	3	0, 7, 17, 27
170307_dance5	dance	Dance	400 s	1	0, 10, 20, 30
170915_office1	office	Others	180 s	1	0, 9, 19, 21
171026_pose3	pose	Range of motion	260 s	1	0, 4, 16, 26

For each sequence, the 3D human pose is provided as ground truth, which can be transformed into the 2D camera frame. This transformation is performed for every camera listed in Table 3 using scripts provided by Panoptic. Additionally, Panoptic keypoints differ from those used by COCO, as shown in Figure 1. Due to these differences, two points (points 0 and 2) are omitted from the ground truth and the remaining keypoints are reordered to maintain consistency with the COCO format.

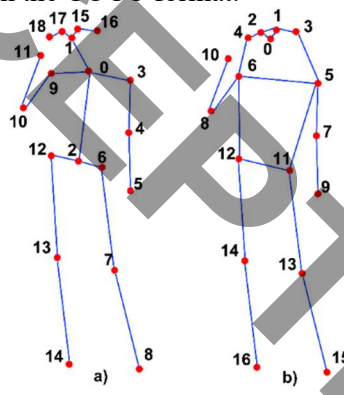


Figure 1. - 2D body keypoint styles: a) Panoptic, b) COCO

For benchmarking purposes, a desktop PC with 64 GB of RAM is used. The PC is equipped with an AMD Ryzen 9 3900X processor featuring 12 cores, 24 threads, and a clock speed of 3.8 GHz. An Nvidia GeForce RTX 2080 SUPER graphics card, with 8 GiB of memory and CUDA version 12.6, is used for computations. The operating system on the PC is Ubuntu 20.04.6 LTS with a Linux kernel version of 5.4.0-202-generic.

5. Evaluation metrics

The Object Keypoint Similarity (OKS) [14] metric is commonly used in pose estimation to assess the alignment of predicted keypoints with the corresponding ground truth keypoints. The formula is given by:

$$\mathrm{OKS} = \frac{\sum_i \exp\left(-\frac{d_i^2}{2s^{2k_i^2}}\right)\delta(v_i > 0)}{\sum_i \delta(v_i > 0)} \tag{3}$$

where d_i is the Euclidean distance between the i-th predicted, and ground truth keypoint, s is the human scale, k_i is a keypoint-specific constant that accounts for variability in localization precision. For example, smaller values of $k_i$are used for less flexible keypoints, v_i is the visibility flag for the ground truth keypoint i, which equals 1 if the keypoint is visible and 0 otherwise, $\delta(v_i > 0)$ is the indicator function that equals 1 if $v_i > 0$, and 0 otherwise.

This metric is particularly useful in multi-person scenarios, where accurately localizing keypoints across varying scales and poses is challenging. However, it has a significant issue regarding the definition of the

ACCEPTED

parameter *s*. Following [14] we approximate it as $s = 0.53w \cdot h$ (width times height) of the bounding box. To ensure fairness in testing, the same ground-truth bounding box was used for all methods. This approach was necessary to ensure a fair comparison, since our evaluation focuses on keypoint localization rather than object detection. Using method-specific bounding boxes would bias the results: inaccurate methods could produce overly large boxes and still obtain artificially high OKS scores despite poor keypoint predictions, due to the scaling factor. Moreover, not all methods output bounding boxes. For these reasons, evaluating all methods under a consistent bounding-box definition is the most appropriate way to compare keypoint localization accuracy.

In multi-person pose estimation, distinguishing between individuals in each frame is essential for comparing predictions with the ground truth. This problem is framed as a classification and assignment task, where the method must estimate the correct number of poses and ensure that keypoints are properly aligned. Properly aligned poses (OKS ≥ *t*) are treated as true positives (TP), excess predictions (unmatched predictions) and misalignments (OKS < *t*) are penalized as false positives (FP), while shortage of predictions (unmatched ground truths) is penalized as false negatives (FN). To do so we construct a weighted bipartite graph between ground truth poses and the predicted poses, with edge weights determined by their respective OKS values. Each ground truth pose is then matched to a predicted pose such that the total weight (sum of OKS values) is maximized. This optimal matching can be computed efficiently using algorithms such as the Hungarian algorithm or Successive shortest path algorithm.

Using the counts of TP, FP, and FN we can compute Precision (also called positive predictive value) and Recall (also known as sensitivity).

$$\text{Precision} = \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

Note that both Precision and Recall depend on the chosen OKS threshold. These values are typically reported as Precision@*t* OKS and Recall@*t* OKS, where *t* is the chosen threshold (e.g. for *t*=0.5 we have Precision@0.5).

Precision and recall are computed per frame, but since large video datasets often contain a substantial number of frames, it is necessary to define aggregate measures for dataset-level performance. These aggregate measures are the Average Precision@*t* OKS (AP@*t*) and Average Recall@*t* OKS (AR@*t*), defined as:

$$\text{AP@}t = \frac{1}{n \cdot m}\sum_{j=1}^{m}\sum_{i=1}^{n}\text{Precision}_{ij}@t\,, \qquad \text{AR@}t = \frac{1}{n \cdot m}\sum_{j=1}^{m}\sum_{i=1}^{n}\text{Recall}_{\text{ij}}@t \tag{5}$$

where *n* is the number of frames in a video and *m* is the number of videos, and accordingly $\text{Precision}_{ij}@t$ and Recall_{ij}@t are Precision and Recall for *i*-th frame and *j*-th video with OKS cutoff set to *t*.

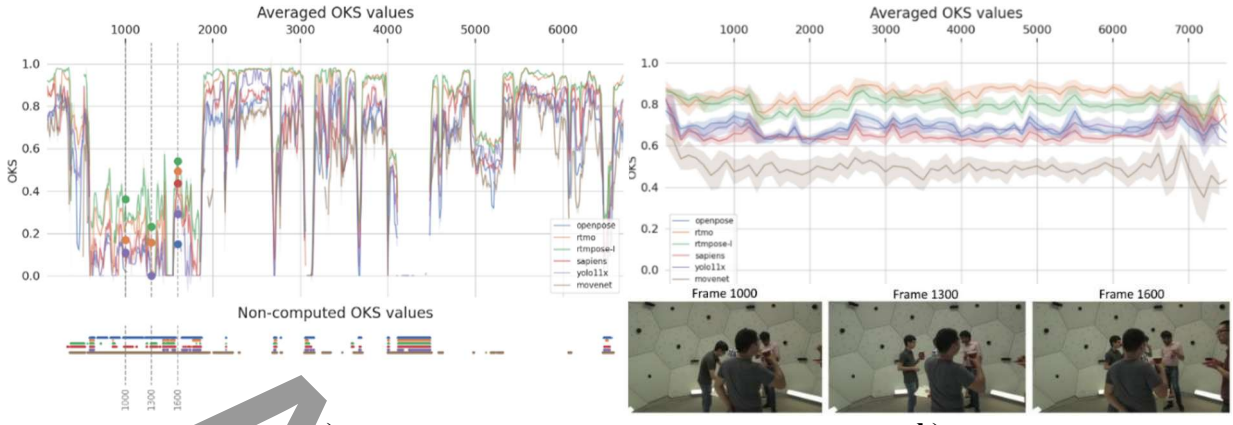
To measure overall performance across the dataset, we compute the mean Average Precision (mAP) and mean Average Recall (mAR), defined as:

$$\text{mAP} = \frac{1}{|T|}\sum_{i \in T}\text{AP@}i, \qquad \text{mAR} = \frac{1}{|T|}\sum_{i \in T}\text{AR@}i \tag{6}$$

where *T* are the threshold we computed over (i.e., 0.5 to 0.95 with 0.05 increment) and |*T*| denotes the number of thresholds.

Note that these metrics depend only on TP, FP, and FN, and do not account for true negatives (TN). Consequently, methods that produce additional poses with very low confidence (e.g., MoveNet) are not penalized.

ACCEPTED



a) **b)**

Figure. 2. – a) OKS of occluded person, b) averaged oks for non-occluded person

ACCEPTED

6. Results

Testing methods under occlusion is crucial for evaluating their robustness. Figure 2 shows the OKS values for two people in the band scenario of camera 6. Figure 2 a) shows the OKS values for an occluded person in a video, specifically the person wearing the checkered shirt. At the beginning of the sequence, this person is clearly visible, and all methods achieve high OKS values. However, around frames 1000, 1300, and 1600, the person becomes occluded, which leads to noticeable drops in OKS, marked by dots in the figure. In some frames, all methods fail to detect the person entirely; these cases are indicated by small dots below the OKS axis in Figure 2 a) and correspond to missed pose estimates despite the presence of ground truth. Figure 2 b) presents a smoothed OKS curve for a different person, where each point represents the average over 10 consecutive frames and the error bars indicate the standard deviation within that window. Some methods exhibit wide bands (e.g., MoveNet), indicating strong frame-to-frame variability. Overall, RTMO and RTMPose maintain relatively high OKS values even when only a small portion of the person is visible, suggesting better robustness to occlusion and more reliable estimation of hidden keypoints. This superior performance can be attributed to their effective estimation of occluded keypoints.Same trend can be seen in Figure 3 and Table 4, where AP and AR were calculated for different OKS thresholds for each method used in this work. The results indicate that the MoveNet (0.37 mAP, 0.71 mAR) achieved the lowest performance in both AP and AR, which was somewhat expected as it is the smallest but fastest method, followed by OpenPose (0.37 mAP, 0.77 mAR). The most accurate results were anticipated from the Sapiens method (0.40 mAP, 0.86 mAR) due to its large model size. However, the Sapiens method slightly underperformed compared to the YOLO11x method (0.43 mAP, 0.89 mAR) in terms of AP and AR, but best results were achieved by the RTMO method (0.61 mAP, 0.95 mAR) for both AP and AR, while the RTMPose method (0.60 mAP, 0.93 mAR) slightly underperformed compared to RTMO.

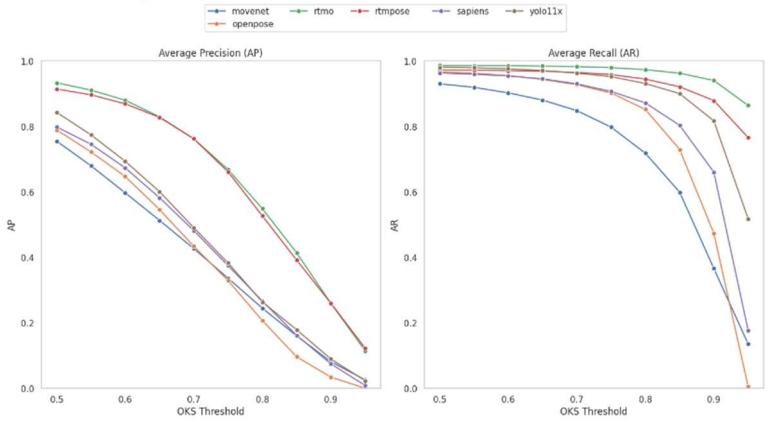


Figure 3 - AP and AR over entire dataset

Table 4. – AP and AR over dataset for used methods, mean ± standard deviation

Method	AP@0.5	AP@0.75	mAP	AR@0.5	AR@0.75	mAR
OpenPose	0.78 ± 0.16	0.31 ± 0.21	0.37 ± 0.28	0.96 ± 0.11	0.87 ± 0.09	0.77 ± 0.02
RTMO	**0.92 ± 0.09**	**0.64 ± 0.18**	**0.61 ± 0.28**	**0.98 ± 0.07**	**0.98 ± 0.05**	**0.95 ± 0.01**
RTMPose	0.89 ± 0.12	0.63 ± 0.20	0.60 ± 0.28	0.97 ± 0.12	0.95 ± 0.07	0.93 ± 0.01
Sapiens	0.78 ± 0.14	0.36 ± 0.18	0.40 ± 0.28	0.97 ± 0.12	0.91 ± 0.09	0.86 ± 0.01
YOLO11x	0.83 ± 0.11	0.37 ± 0.19	0.43 ± 0.28	0.96 ± 0.09	0.94 ± 0.06	0.89 ± 0.01
MoveNet	0.74 ± 0.20	0.32 ± 0.23	0.37 ± 0.25	0.90 ± 0.17	0.75 ± 0.16	0.71 ± 0.02

Figure 4 compares the average processing time per sequence with each video's duration marked by the black line. In terms of speed and hardware efficiency, Sapiens was predictably the slowest at 0.6 FPS due to its large model, with OpenPose next at 10.7 FPS, which still falls short of real-time performance. On the other hand, MoveNet was the fastest at 72.48 FPS. On multi-person scenarios, YOLO11x ran substantially faster, though it was slightly behind RTMPose on single-person clips, including the *ian* scenario featuring a toddler and mother. Overall, RTMPose outperformed RTMO, except in the most crowded scene where RTMO's bottom-up strategy had the advantage.

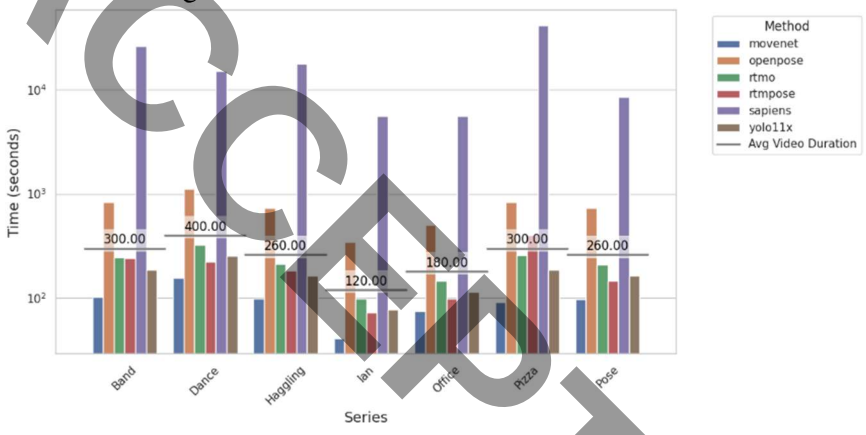


Figure 4. - Average processing time over cameras for different video sequences

As shown in Table 5, MoveNet stands out for its extremely low hardware utilization, using only 10-15% of GPU, about 1 GiB of GPU memory, 2.2 GiB of RAM, and 1.5 CPU cores. This makes MoveNet particularly suitable for deployment on resource-constrained devices, such as mobile and edge hardware. However, this efficiency comes at the cost of lower accuracy compared to the other evaluated methods, cf. RTMO and RTMPose which give good results but have moderate hardware demands.

Table 5. – Speed and hardware utilization

Method	GPU %	GPU memory	CPU %	RAM usage	FPS
OpenPose	87-91	4.340/8 GiB	175-184	3.16/62.8 GiB	10.7
RTMO	80	1.696/8 GiB	420-440	5.93/62.8 GiB	36.7
RTMPose	65-70	2.041/8 GiB	564-590	6.11/62.8 GiB	44.9
Sapiens	90-95	3.469/8 GiB	400-520	5.07/62.8 GiB	0.6
YOLO11x	80	**0.612/8 GiB**	165-172	3.70/62.8 GiB	47.8
MoveNet	**10-15**	1.024/8 GiB	**140-155**	**2.21/62.8 GiB**	**72.48**

7. Conclusion

In this paper, a detailed benchmarking process was conducted for six methods of 2D human pose estimation: OpenPose, RTMO, RTMPose, Sapiens, YOLO11x, and MoveNet. The evaluation utilized 28 sequences from the Panoptic dataset, totaling 121.33 minutes at 30 FPS in HD resolution. This dataset provides 3D ground-truth poses for each person within the dome for a given scene, enabling accurate evaluation and comparison of the methods under diverse conditions.

ACCEPTED

For each method evaluated in this work, the OKS was calculated, from which precision and recall were derived for each frame in the video. Based on precision and recall, AP, AR, mAP and mAR metrics were computed. RTMO achieved the highest overall mAP/mAR (0.61 mAP, 0.95 mAR) on the selected benchmark and also showed strong behaviour in the analysed occlusion examples. Conversely, the most precise method was not the fastest. The MoveNet method achieved the fastest results (72.48 FPS) while utilizing the least hardware resources (2.21 GiB RAM, 140-155 CPU%, 10-15 GPU%), second only to YOLO11x (0.612 GiB GPU) in GPU consumption. These results highlight the trade-offs betweesen precision, speed and resource usage. In scenarios involving many people, RTMO is the preferred option because of its bottom-up approach, which handles crowded scenes effectively, while RTMPose is faster in scenes of up to four people. In resource-constrained tasks where moderate AP is acceptable, MoveNet is best due to its high speed and minimal hardware usage.

Ultimately, the best method is the one that fulfills the specific needs of the user for a given application, balancing precision, speed, and hardware constraints.

ACCEPTED

Acknowledgment

This work was supported by the National Recovery and Resilience Plan 2021-2026 (NPOO) under the project “Advanced Algorithms and Optimization Models Supported by Mathematical Theory – OptimaAI” (581-UNIOS-54).

References

[1] The graphic: Semi-automated offside system. Engineering & Technology, 2022.
[2] Tewodros Legesse Munea, Yalew Zelalem Jembre, Halefom Tekle Weldegebriel, Longbiao Chen, Chenxi Huang, and Chenhui Yang. The progress of human pose estimation: A survey and taxonomy of models applied in 2d human pose estimation. IEEE Access, 8:133330–133348, 2020.
[3] Haoming Chen, Runyang Feng, Sifan Wu, Hao Xu, Fengcheng Zhou, and Zhenguang Liu. 2d human pose estimation: a survey. Multimedia Systems, 2023.
[4] Qi Dang, Jianqin Yin, Bin Wang, and Wenqing Zheng. Deep learning based 2d human pose estimation: A survey. Tsinghua Science and Technology, 2019.
[5] Sakshi Kulkarni, Shubham Deshmukh, Favin Fernandes, Aniket Patil, and Vaishali Jabade. Poseanalyser: A survey on human pose estimation. SN Computer Science, 2023.
[6] Gongjin Lan, Yu Wu, Fei Hu, and Qi Hao. Vision-based human pose estimation via deep learning: A survey. IEEE Transactions on Human-Machine Systems,2023.
[7] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: Realtime multi-person 2d pose estimation using part affinity 2021.
[8] Glenn Jocher and Jing Qiu. Ultralytics yolo11, 2024.
[9] Peng Lu, Tao Jiang, Yining Li, Xiangtai Li, Kai Chen, and Wenming Yang. RTMO: Towards High-Performance One-Stage Real-Time Multi-Person Pose Estimation . CVPR, pages 1491-1500, 2024.
[10] Tao Jiang, Peng Lu, Li Zhang, Ningsheng Ma, Rui Han, Chengqi Lyu, Yining Li, and Kai Chen. Rtmpose: Real-time multi-person pose estimation based on mmpose, 2023.
[11] Rawal Khirodkar, Timur Bagautdinov, Julieta Martinez, Su Zhaoen, Austin James, Peter Selednik, Stuart Anderson, and Shunsuke Saito. Sapiens: Foundation for human vision models, 2024.
[12] MoveNet: Ultra fast and accurate pose detection model. TensorFlow Hub - tensorflow.org. https://www.tensorflow.org/hub/tutorials/movenet. [Accessed 16-07-2025].
[13] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. ICCV, pages 3334-3342, 2015.
[14] Gabriela Vdoviak and Tomyslav Sledevič. Enhancing keypoint detection in thermal images: Optimizing loss function and real-time processing with yolov8n-pose. AIEEE, pages 1-5, 2024